A Circuit Representation Technique for Automated Circuit Design

Jason D. Lohn, Silvano P. Colombano

Abstract

We present a method of automatically generating circuit designs using evolutionary search and a set of circuit-constructing primitives arranged in a linear sequence. This representation has the desirable property that virtually all sets of circuit-constructing primitives result in valid circuit graphs. While this representation excludes certain circuit topologies, it is capable of generating a rich set of them including many of the useful topologies seen in hand-designed circuits. Our system allows circuit size (number of devices), circuit topology, and device values to be evolved. Using a parallel genetic algorithm and circuit simulation software, we present experimental results as applied to three analog filter and two amplifier design tasks. In all tasks, our system is able to generate circuits that achieve the target specifications. Although the evolved circuits exist as software models, detailed examinations of each suggest that they are electrically well behaved and thus suitable for physical implementation. The modest computational requirements suggest that the ability to evolve complex analog circuit representations in software is becoming more approachable on a single engineering workstation.

Keywords

automated design, analog circuit synthesis, genetic algorithms, circuit representation

I. Introduction

Analog circuits are of great importance in electronic system design since the world is fundamentally analog in nature. While the amount of digital design activity far outpaces that of analog design, most digital systems require analog modules for interfacing to the external world. It was recently estimated that approximately 60% of CMOS-based application-specific integrated circuit (ASIC) designs incorporated analog circuits [1]. With challenging analog circuit design problems and fewer analog design engineers, there are economic reasons for automating the analog design process, especially time-to-market considerations.

Techniques for analog circuit design automation began appearing about two decades ago. These methods incorporated heuristics [18], knowledge-bases [5], and simulated annealing [15]. Efforts using techniques from evolutionary computation have appeared over the last few years. These include the use of genetic algorithms (GAs) [6] to select filter component sizes [7], to select filter topologies [3], and to design operational amplifiers using a small set of topologies [13]. The research of Koza and collaborators [11] on analog circuit synthesis by means of genetic programming (GP) [10] is likely the most successful evolutionary computation-based approach to date. Unlike previous systems, the component values, number of components, and the circuit topologies are evolved. The genetic programming system begins with minimal knowledge of analog circuit design and generates circuits using a cellular encoding technique [4] and circuit-constructing program trees. Various analog filter design problems have been solved using genetic programming (e.g., [12]), and an overview of these techniques, including eight analog circuit synthesis problems, is found in [11]. A comparison of genetic-based techniques applied to filter design appears in [19] and work on evolving CMOS transistors for function approximation [17] has also appeared recently.

The evolutionary computation-based approaches mentioned above and described in this paper are all *offline* hardware evolution approaches – software simulators are used to evolve software models of hardware. It should be noted that a growing number of researchers are using real hardware to evaluate a circuit's fitness *during* evolutionary search – the *online* approach. For example, in this issue, a system that employs a software-reconfigurable analog circuit to control frequency filtering in

Manuscript received on ___. This work was supported by the NASA New Horizons Research Program.

J. D. Lohn is with Caelum Research Corporation at NASA Ames Research Center, Mail Stop 269-1, Moffett Field, CA 94035-1000 (email: jlohn@ptolemy.arc.nasa.gov).

S. P. Colombano is with the Computational Sciences Division of NASA Ames Research Center, Mail Stop 269-1, Moffett Field, CA 94035-1000 (email: scolombano@mail.arc.nasa.gov).

cellular phones is reported [14]. They demonstrate how genetic algorithms are used to shrink circuit sizes and improve manufacturing yield. Another study that uses online evolution (and also appears in this issue [8]) investigated the use of evolutionary search to make a field-programmable gate array (FPGA) function like an oscillator. Oscillators are difficult to design using only digital logic gates, so in practice they are made using more expensive analog components. Since FPGAs are digital, the design task is challenging. The authors reported results from 30 genetic algorithm runs that produced accurate oscillators for half of the target frequencies.

Our investigation centers on whether a linear representation and unfolding technique, coupled with modest computer resources, could be effective for evolving analog filters and transistor-based amplifiers. Our circuit representation has the desirable property that all sets of circuit-constructing primitives result in valid circuit graphs, with the exception of a few trivial cases. Thus application of genetic operators will result in valid circuit graphs. While certain circuit topologies are excluded, our representation is capable of generating a rich set of them including many of the useful topologies seen in hand-designed circuits. Another motivation in designing our representation was to minimize the amount of computer processing and hence run time of our system. Since virtually all of the graph structures generated are valid circuits, our system does not need to perform functions such as the pruning of unconnected circuit branches.

Our technique presented below differs from the previously mentioned GA techniques in that we allow both topology and component sizes to be evolved. In [19], a GA approach is presented in which topologies and component values are evolved, however that system allowed only a small number of components (15 components maximum). Because our genome resembles a computer program, and because our GA acts upon dynamically-sized representations, it has some elements in common with genetic programming. Our genomes are fixed to hold a maximum of 150 circuit components, a limit arrived at by surveying the circuit design literature for the design tasks we are interested in. Using a cluster of six engineering workstations (1996 Sun Ultra), we present evolved circuit solutions to four circuit design tasks. Although the evolved circuits exist as software models, detailed examinations of each suggest that they are electrically well-behaved and thus suitable for physical implementation.

The remainder of this paper is organized as follows. The circuit representation technique is introduced and described in Section II. Section III presents the evolutionary computation, and in Section IV our experimental results are presented and analyzed. We discuss our conclusions in Section V.

II. Circuit Representation

In designing an effective circuit representation for use in evolutionary search, the following properties are among the most desirable. First, the representation should permit any circuit or at least a wide range of circuits to be represented. If it is known *a priori* that certain topologies are well suited to a specific design task, topological restrictions inherent in the representation may be beneficial since the search space will be reduced. Conversely, not having this limitation may bring to light novel designs that human designers have never envisioned. Second, the genotype conversion algorithm (the circuit constructing process) should run as fast as possible. Clearly if numerous traversals of the circuit graph structure are required in order to guarantee a valid circuit graph, the performance hit will be commensurate. For an n-component circuit, a reasonable upper bound would be $O(n)$. Third, the representation should be syntactically closed so that genetic operators do not create invalid circuit graphs[1] from those that are valid. The circuit representation we present here was designed to have these properties.

Circuit designs are constructed by an automaton that is programmed via a set of low-level instructions. The automaton is called a circuit-constructing robot or cc-bot, and the "language" that programs it is small and, in its current incarnation, contains only component-placing instructions (e.g., control instructions are not included). The set of cc-bot instructions has the desirable property that virtually all possible sequences of instructions result in a valid electrical circuit. This property is important because it greatly limits the "out-of-bounds" regions of the search space containing invalid circuit graphs. Thus, evolutionary search will spend nearly all its time generating valid circuit graphs. While this is a beneficial, non-trivial achievement, we do lose the ability to generate every possible circuit topology. This is not considered a drawback for the circuit types we investigated since a vast number of topologies and existing circuit designs could be encoded using the cc-bot approach.

Each cc-bot instruction places a circuit component and directs the movement of the cc-bot. The

[1]Note that a graph could be a valid circuit graph, yet not make sense as an electrical circuit – for example, dissimilar voltage sources connected in parallel.

five basic instruction types are: x-move-to-new, x-cast-to-previous, x-cast-to-ground, x-cast-input, x-cast-to-output, where x can be replaced by R (resistor), C (capacitor), L (inductor), or transistor configuration. In a circuit design task involving only inductors and capacitors (an LC circuit), ten opcodes would be available to construct circuits (five for capacitors and five for inductors).

The meanings of each instruction are summarized in Table I. The move-to-new instruction places one end of a component at the active node and the other at a newly created node (the "active" node is the current location of the cc-bot). The newly created node then becomes the active node. The cast-to instructions place one end of the component at the active node and the other at either the ground, input, output, or previously-created node. After executing a cast-to instruction, the cc-bot remains at the active node. The input and output nodes are the overall input and output nodes of the circuit as opposed to the input and output of the placed component. Illustrations of two instructions that place resistors are shown in Fig. 1.

Instruction	Outgoing Node	Active Node
x-move-to-new	newly-created node	becomes the newly-created node
x-cast-to-previous	previous node	unchanged
x-cast-to-ground	ground node	unchanged
x-cast-to-input	input node	unchanged
x-cast-to-output	output node	unchanged

TABLE I
SUMMARY OF OPCODE TYPES USED IN CURRENT SYSTEM. x DENOTES THE COMPONENT TYPE: RESISTOR, CAPACITOR, INDUCTOR, OR TRANSISTOR CONFIGURATION.

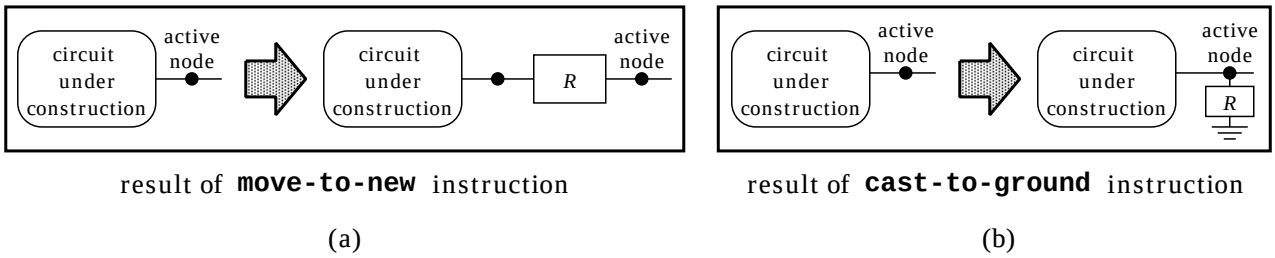

Fig. 1. Effect of placing a resistor with (a) move-to-new, and (b) cast-to-ground instructions.

The circuit is constructed by the cc-bot inside of a template circuit. The design tasks presented here use a template having one input and one output terminal as shown in Fig. 2. An ideal voltage source v_s is connected to ground and to a source resistor R_s. The circuit's output voltage is taken across a load resistor R_l.

The lists of cc-bot instructions manipulated by the GA are variable-length lists so that the size of the circuit can be evolved. When the cc-bot reaches the last component to place in the circuit, we arbitrarily chose to have the last active node connected to the output terminal by a wire (accomplished by connection of a $1\mu\Omega$ resistor). By doing so, we eliminate unconnected branches.

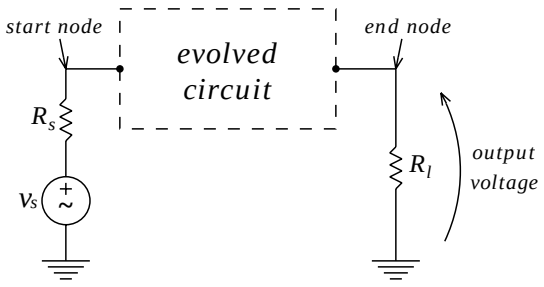

Fig. 2. Template circuit: the evolved circuit is located between fixed input and output terminals. v_s is an ideal voltage source, R_s is the source resistance, R_l is the load resistance.

As assembly language instructions are mapped to opcodes, cc-bot instructions are mapped to bytecodes. Instructions are represented by up to four bytecodes. For instructions that take a component value as an argument, the first byte is the instruction, and the next three represent the component value (resistance, capacitance, and inductance values). For transistors, component values are not needed. Using three bytes allows the component values to take on one of 256^3 values, a sufficiently fine-grained resolution. The raw numerical value of these bytes was then scaled into a reasonable range, depending on the type of component. Resistor values were scaled sigmoidally between 1 and 100K ohms using $1/(1+\exp(-1.4(10x-8)))$ so that roughly 75% of the resistor values were biased to be less than 10K ohms. Capacitor values were scaled between approximately 10 pF and 200 μF and inductors between roughly 0.1 mH and 1.5 H.

Transistors are current amplifying and switching devices that have three terminals.[2] In this paper we use bipolar junction transistors as shown in Fig. 3. Using devices with three terminals makes it harder to design a circuit representation that achieves the properties that we desire. If a cc-bot were to connect one terminal of a transistor at a node, then two active nodes would result each requiring its own cc-bot. This could happen repeatedly resulting in an exponential growth of cc-bots constructing the circuit in parallel. Two problems are obvious: how will the multiple constructing "threads" interconnect, and how will the dangling nodes that will likely appear at the end of the circuit constructing process be handled. To allow interconnections between constructing threads, one can introduce a spatial dimension and let the cc-bots form interconnections as they criss-cross each other's path. To handle the dangling node problem, one can determinstically tie dangling nodes to each other, to internal nodes, or to the output node for example. Another solution is to simply prune those nodes. Although we have considered these and other solutions, a simpler way of handling transistors is also a viable alternative: treating them as having only two terminals.

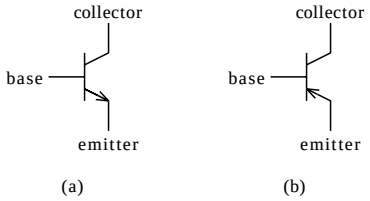

Fig. 3. Bipolar junction transistor symbols: (a) npn; (b) pnp.

To work with transistors as devices with two terminals, we have the third terminal hardwired (fixed) to one of the following pre-existing circuit nodes: ground, power supply (positive or negative), input, output, the previously placed node, or even to itself. Such a scheme allows a wide variety of configurations. To understand these configurations we label the terminals in a generic way: incoming, outgoing, and fixed (see Fig. 4). The incoming terminal is the terminal that the cc-bot will connect to the active node. The outgoing terminal will become the new active node (for move-to instructions) or it will be cast to a pre-existing circuit node. The fixed terminal is hardwired as its name implies.

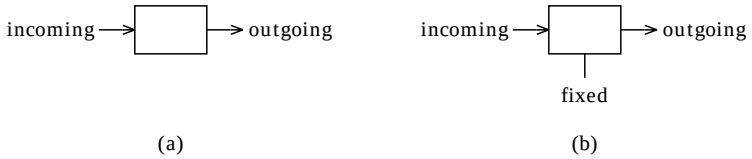

Fig. 4. Labeling of terminals: (a) devices with two terminals have incoming and outgoing terminals; (b) devices with three terminals are treated as two-terminal devices by having a fixed connection at the third terminal.

To give a sense of the types of transistor configurations possible, the chart in Fig. 5 illustrates 52 configurations for an npn transistor whose base terminal is designated incoming. Each entry shows the connections that the cc-bot makes when executing the instruction listed in the first column. The last two columns show self-connections, some of which are frequently used by circuit designers. Similar charts can be produced for npn transistors having the collector and emitter serve as the incoming terminal, as well as the three analogous charts for pnp transistors. There are configuration redundancies so that each chart will not have exactly 52 configurations. In addition we exclude emitter-collector self-connections since this shorts out the transistor.

[2]Four if the substrate terminal is included, but we connect the substrate to ground and hence ignore it.

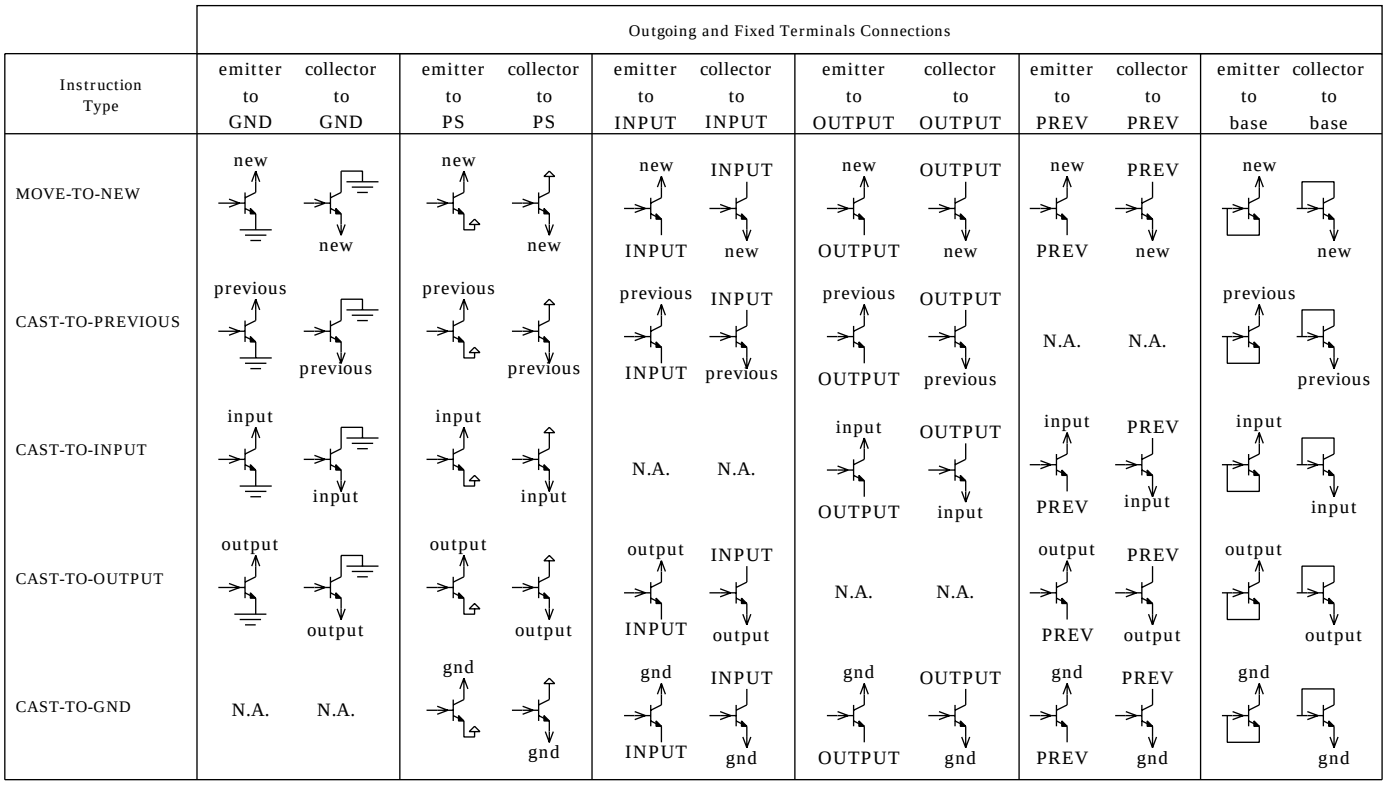

Fig. 5. Transistor configurations showing the outgoing and fixed connections when the incoming terminal is the transistor's base terminal. Terminals labeled in upper case letters denote the fixed terminal connection. In the last two columns the fixed connection is a self-connection. "PS" denotes the power supply (only the positive version is shown), and "N.A." denotes "not applicable." Only npn transistors are shown although analogous configurations are present for pnp transistors.

The cc-bot approach to representing circuits embodies the desirable properties outlined above. The encoding has syntactic closure since any combination of instructions produces a valid circuit graph, and since every instruction contained in the genome results in a circuit component, there are no non-coding genome segments. The circuit construction process is $O(n)$ since it does not require any repair (e.g., removal of unconnected nodes) operations. Lastly, the cc-bot approach can generate a wide range of circuit graph topologies. The topological restriction is as follows: circuit branches off of the main constructing thread cannot, in general, contain more than one node (there are some exceptions to this). The constructing thread is the sequence of components that are created by the move-to-new instructions. The constructing thread itself can be of varying lengths and can contain both series and parallel configurations. In spite of these limitations, our system allows the creation of circuits with a large variety of topologies, including numerous topologies seen in hand-designed circuits.

III. Evolutionary Search

The evolutionary search employed in our experiments is based on the genetic algorithm. The GA operates on a population of dynamically-sized bytecoded arrays. In practice we imposed a maximum size of about 400 bytes (100-150 circuit components) in order to accommodate population sizes of up to 18,000 individuals. The crossover rate was set at 0.8 and mutation (per locus) rates were set between 0.05-0.20. Crossover was single-point with each parent having a separately chosen crossover point. Crossover points were randomly selected and were constrained to lie on component boundaries and to yield circuits having at least 10 components and at most 150 components.

An overview of the evaluation process is depicted in Fig. 6. As in the GP system mentioned above, we used the public-domain Berkeley SPICE (Simulation Program with Integrated Circuit Emphasis) circuit simulation program [16] to simulate our circuits. The array of bytecodes was interpreted in the manner previously described, and resulted in a SPICE netlist representation. The netlist is processed by SPICE and the output from SPICE is then used to compute fitness for the individual.

Since most of the processing time is spent simulating circuit designs, which are independent of

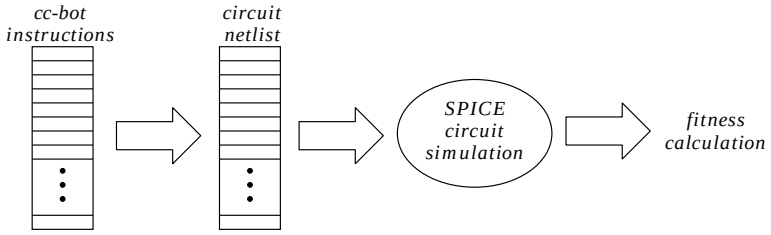

Fig. 6. Overview of circuit evaluation process starting with cc-bot instructions and ending with calculation of fitness.

one another (within a generation), parallelizing the system is greatly beneficial. The parallel GA implemented uses master/slave-style parallelism [2] over a network of UNIX-based workstations. A controlling host computer performs GA operators and distributes a population of bytecoded-individuals to a specified number of worker nodes using socket connections (Fig. 7). The worker nodes decode the individuals into SPICE netlists that are then fed into SPICE via FIFO pipes to minimize disk activity. Fitness is calculated using SPICE's output, and then sent back to the host. Hundreds of individuals (and fitness scores) are packaged into a single message so that external network congestion delays are minimized. The SPICE program itself required minor modifications since it runs as a separate process. Written in the C programming language, the system currently runs on Sun workstations and a variety of other UNIX systems (e.g., SGIs, PCs running Linux), giving us the ability to mix various workstations in a single workstation cluster.

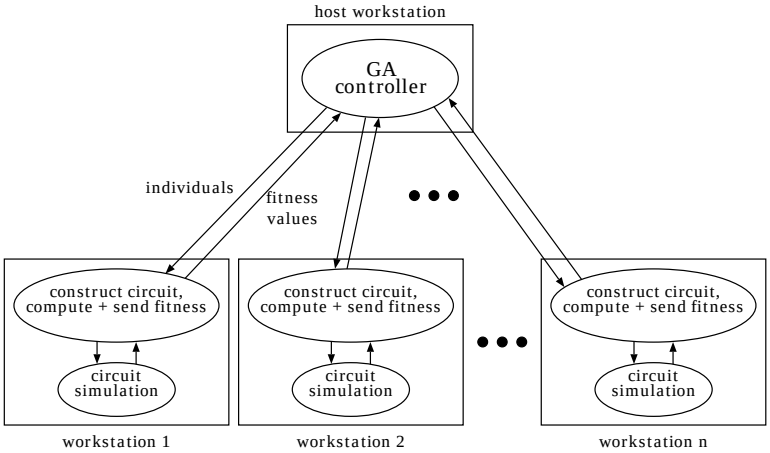

Fig. 7. Parallelization of genetic algorithm. Running circuit simulations in parallel greatly reduces runtime since nearly all of the processing time is spent simulating circuits.

IV. Experiments and Results

The goal of the experiments was to automatically design analog filters and transistor-based amplifiers using evolutionary search. The choice of using passive analog filters was inspired by the previous studies and is a good choice for testing the effectiveness of our system for three reasons. First, all components have two terminals which is suited for the cc-bot technique. If the proposed system could not evolve useful circuits using devices with two terminals, then attempting to evolve circuits using more complex components (e.g., transistors) would likely prove ineffective. Second, there are no energy sources required within the circuit which further reduces the complexity. Lastly, filter design is a well-understood discipline within circuit design. Its "design space" has been greatly explored [9] which allows us to compare our evolved designs to well-known designs. Amplifiers are very common in analog circuit design and are a good choice for our experiments for many of the same reasons that filters are. We use transistors in the amplifier design experiments since the vast majority of such designs today are transistor-based.

All of the evolved designs presented below meet the target specifications. For filters, that meant satisfying four parameters. For the amplifier circuits, our gain target was a range of values: from 70 dB to the maximum gain attainable for the circuit. Each evolved circuit was hand-probed using

simulation software so that we could verify that internal currents and voltages remained at realistic values. This step suggests that these circuits could be physically implemented.

A low-pass filter is a circuit that allows low frequencies to pass through it, but stops high frequencies from doing so. In other words, it is frequency selective in that it "filters out" frequencies above a specified frequency. The unshaded area in Fig. 8 depicts the region of operation for low-pass filters. Below the frequency f_p the input signal is passed to the output, potentially reduced (attenuated) by K_p decibels (dB). This region is known as the passband. Above the frequency f_s, in the region is called the stopband, the input signal is markedly decreased by K_s decibels. Between the passband and stopband the frequency response curve transitions from low to high attenuation. The parameter located in this region, f_c, is known as the cutoff frequency.

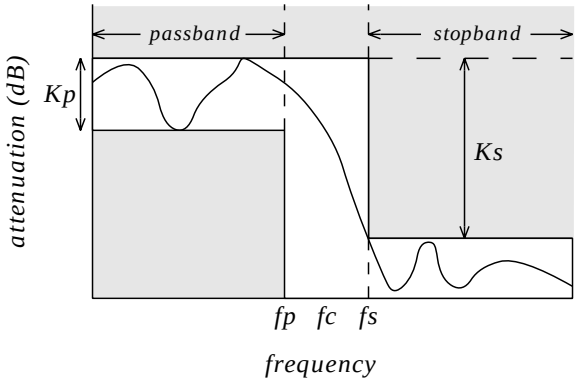

Fig. 8. Low-pass filter terminology and specifications. The shaded regions represent out-of-specification areas. An example frequency response curve that meets specifications is shown.

One of the design tasks concerned designing a circuit within the class of "Butterworth" filters. Butterworth filters are very common and circuits that implement them are readily found in filter design tables [9]. The attenuation (negative gain) of Butterworth filters is of the form $\sqrt{1/(1+(f/f_c)^{2N})}$ where f is the input frequency and N is the order of the filter. The higher order a filter has, the sharper the "knee" of its gain curve, and the more complex the circuit. A plot of the attenuation for a third-order Butterworth filter is shown in Fig. 9.

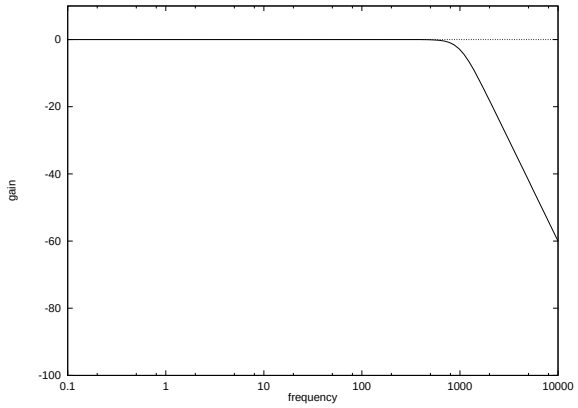

Fig. 9. Gain on a logarithmic amplitude scale for a third-order Butterworth filter.

Amplifier circuits generate an output signal that consists of the input signal multiplied by a gain factor, A. Voltage gain is denoted A_v and is equivalent to the ratio of output to input voltages (v_o/v_i). It is common to express gain values in decibels (dB) using $20\log_{10}(A)$. Amplifiers may be either inverting or non-inverting, where an inverted output signal has a 180 degree phase shift compared to the input. The dc gain of an amplifier refers to the gain when only constant voltage/current sources are applied. The ac or small-signal gain is the gain of an ac input signal irrespective of any dc components in the output signal. A dc component that shifts the entire ac signal up or down is called the dc bias of the circuit. For simple amplifiers, like filters, there are published volumes available that catalog many designs. Since there are numerous parameters in amplifier design (e.g., input/output

impedance, power dissipation, distortion, common-mode rejection, power supply rejection), the design task can become quite challenging and may require an experienced designer. For the amplifier design experiments below, we take into account only dc and ac gain, dc bias, and linearity, although future experiments will include more design parameters.

A. Filter Design Tasks

Three filter design experiments were performed. In each experiment, 10 runs were performed and we present the circuit having the highest fitness value across all runs. The experiments increased in difficulty so that filter 3 represents a challenging design task, while filter 1 is least challenging. Table II lists the target specifications for each of the experiments.

Filter No.	f_p (Hz)	f_s (Hz)	K_p (dB)	K_s (dB)
1	100	4000	1.29	27.12
2	925	3200	3.01	22.00
3	1000	2000	0.01	63.50

TABLE II
TARGET SPECIFICATIONS FOR FILTER DESIGN TASKS.

The GA parameters remained the same within a given experiment, but varied in the number of evaluations (circuit simulations): filter 1 runs had 30,000 evaluations, filter 2 had 3.6 million, and filter 3 had 1 million. These values were arrived at by experimentation and constrained by practical issues such as the availability of workstations.

For the filter experiments, fitness was calculated to promote the regression of the evolved circuit's frequency response toward that of the target. Error values were computed as the absolute value of the difference of the individual's output and the target output. These error values were summed across evaluation points to arrive at a fitness value.

A.1 Electronic Stethoscope Circuit

The first filter design task was set up to generate a filter suitable for use in an electronic stethoscope. In this application, it is desired to filter out the extraneous high-frequency sounds picked up by a microphone which make it difficult to listen to (low-frequency) bodily sounds (e.g., a heart beating). As such, the frequency response specifications do not need to be extremely accurate since the human ear cannot discern frequencies that are close together. The target frequency response data was taken from an actual electronic stethoscope, which was built with a cutoff frequency of 796 Hz corresponding to an output voltage of approximately 1 volt. This circuit is relatively easy to design and so we chose it as our first design task. The cc-bot instruction set consisted of ten instructions, five for resistors and five for capacitors, which allowed for the construction of an RC low-pass filter. The evolved circuit is shown in Fig. 10(a) and its frequency response, which matches almost exactly the target is shown in Fig. 10(b). It was found in generation 3 of a 10-generation run that had a population size of 3000, an indication that this design task was relatively easy. The circuit exhibits the standard design for simple low-pass filters: a resistor (R_2) in series with the source to form a voltage divider at low frequencies (C_1 open), and a capacitor (C_1) across the output to short it at high frequencies.

A.2 Butterworth Low-pass Filter

The second low-pass filter design task had specifications that were more difficult to achieve than the first filter: both the passband and the stopband were longer, thus requiring the transition to be sharper. We chose a circuit that can be built using a 3rd-order Butterworth filter and having a frequency response of the form seen in Fig. 9. The specifications are listed under filter number two in Table II.

Such a filter design can be derived using a ladder topology containing two capacitors and one inductor and component values found in published tables. Because we wanted to design an LC low-pass filter, the cc-bot instruction set consisted of only capacitor and inductor instructions. The evolved circuit that meets these specifications is shown in Fig. 11 and its frequency response is shown in Fig. 12. It was found in generation 22 of a run that had a population size of 18,000.

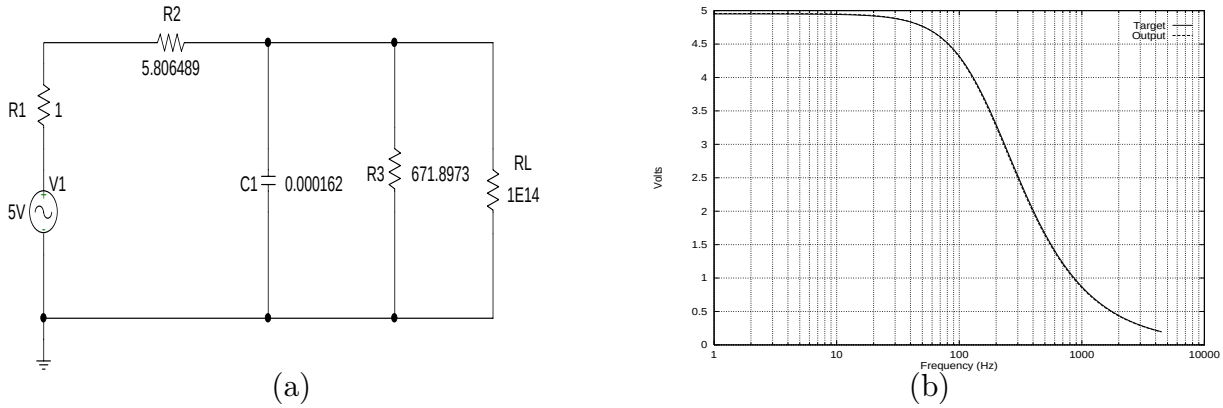

Fig. 10. (a) Evolved low-pass filter for use in an electronic stethoscope (units are ohms and farads); (b) Nearly identical frequency response curves for evolved and actual electronic stethoscope circuit. The frequency axis is scaled logarithmically.

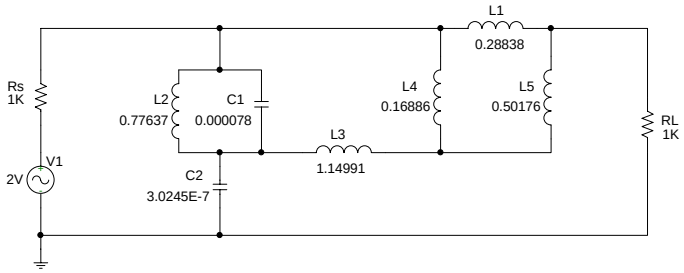

Fig. 11. Evolved 3rd-order Butterworth low-pass filter (units are ohms, farads, and henries).

A.3 Third Low-pass Filter

The third low-pass filter design task had specifications that were the most stringent: in addition to the passband and stopband being increased, the attenuation parameters were tightened (see Table II). These specifications are similar to the fifth-order elliptic filter described in [11]. In that work, the evolved LC circuit satisfies $K_p = 0.3$ dB and $K_s = 60$ dB. Another evolved low-pass filter circuit [19] had the same stopband and passband frequencies, but less demanding attenuation specifications ($K_p =$ 1.6 dB and $K_s = 24.8$ dB). The evolved circuit is shown in Fig. 13 and its frequency response is seen in Fig. 14. Micro-ohm resistors were added as a convergence aid for the circuit simulator, and can be ignored for analytical purposes. This circuit was found in generation 997 of a run that had a population size of 1000. The cc-bot instruction sequence and SPICE netlist for this circuit are included for reference in the Appendix.

B. Amplifier Design Tasks

Two amplifier design experiments were performed. In each experiment, 10 runs were performed and we present the highest performance circuits found across all runs. The goal was to design an inverting amplifier capable of a dc voltage gain up to a maximum of either 100 dB or 120 dB, while minimizing dc bias and maximizing linearity over the dc gain. Population size was set to 1200 individuals, and each run proceeded for 5000 generations, giving a total of 6 million circuit evaluations per run. The difference between the two sets of experiments is that in the first set, the maximum gain was set to be 120 dB, and in the second set, 100 dB was the maximum gain. The maximum gain possible is set by using feedback resistors (labeled R_{FB}). For an ideal inverting amplifier (as shown in Fig. 15), the magnitude of the gain of the amplifier is simply R_{FB}/R_S, where R_S is the source resistor. Fitness was calculated in a manner similar to the work on amplifiers in [11]. An error value is computed as the sum of the dc gain penalty (the target gain minus the observed gain), the dc bias (zero dc bias is ideal), and the degree to which the dc gain is linear.

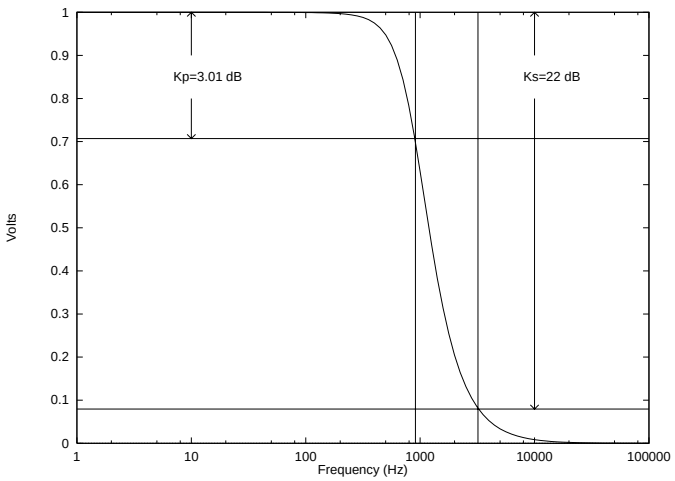

Fig. 12. Frequency response curve for evolved 3rd-order Butterworth low-pass filter. Attenuation specifications are also shown. The frequency axis is scaled logarithmically.

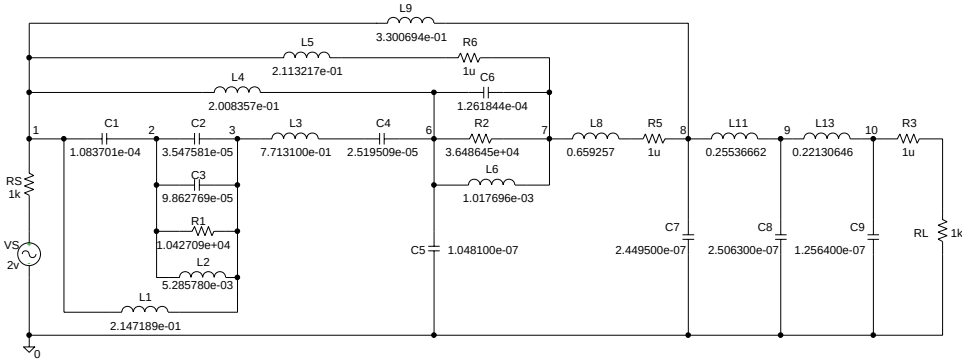

Fig. 13. Evolved circuit satisfying target specifications for filter number three.

B.1 75 dB Inverting Amplifier

In the first set of experiments the maximum voltage gain was set at 120 dB (10^6). The amplifier having the best performance had a dc gain of 74.53 dB (5324.40). Figure 16 shows the schematic for this circuit. It was found in generation 4866, and had a dc bias of 3.64 volts and a power dissipation of 0.82 watts.

The dc behavior is best understood by examining the major current pathways in the circuit. The current through the load is the key quantity since it is converted to a voltage by the load resistor and hence forms the circuit's output. Nearly all of the dc current flowing through the load resistor originates from the power supply connected to transistor Q7's collector. Q7 is biased in such a way as to supply Q8's base with approximately 36.4 mA of current. This current is divided so that 18.1 mA flows out of Q8's emitter and 18.3 mA out of Q8's collector. Resistor R2 is a tiny resistance that was positioned in order to connect transistor Q9 to the output (the last component is forced to connect to the output terminal). Thus R2 can be ignored, and transistor Q9's 18.4 mA current flows into the output node. Currents are summed at node 255 to give the load current of 36.4 mA which flows through the load resistance to give 3.64 volts output. Because there is a negligible amount of current flowing through transistors Q1 through Q4, the utility of these transistors is unclear. Components that are essentially non-functional, are quite commonly seen in evolutionary design applications. Figure 17(a) shows the time domain response. Amplification of a 1 kHz sine wave having a 1 microvolt amplitude can clearly be seen. Figure 17(b) shows the frequency response. The ac gain remains flat at 74.36 dB until it loses 3 dB at 7.59 kHz (its 3 dB bandwidth). Figure 18 shows the dc transfer characteristic. The dc bias of 3.64 volts can be seen at the voltage input of zero volts.

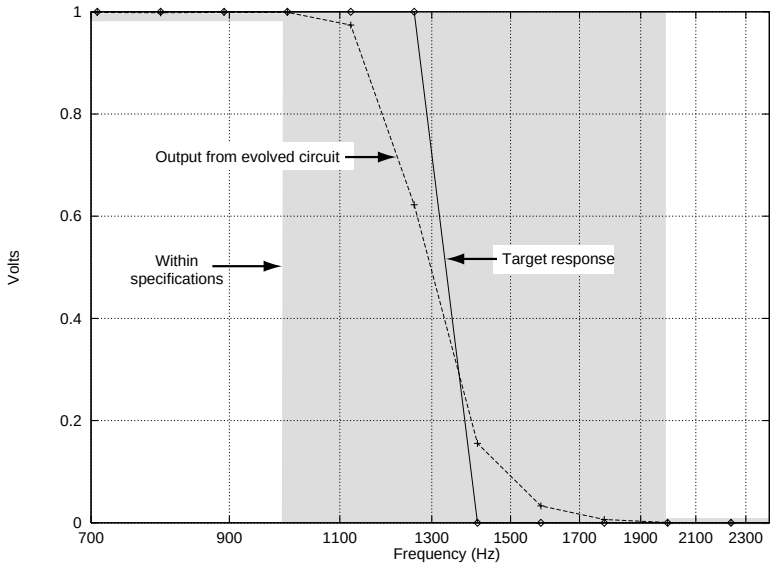

Fig. 14. Frequency response for filter number three.

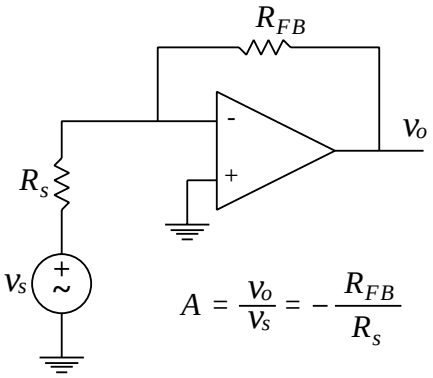

Fig. 15. Ideal inverting amplifier showing how gain is set by the ratio of the feedback to source resistor.

B.2 85 dB Amplifier

In the second set of amplifier experiments the maximum voltage gain was set at 100 dB (10^5). The amplifier having the best performance had a dc gain of 85.41 dB (18, 642.33). Figure 19 shows the schematic for this circuit. It was found in generation 3635, and had a dc bias of 5.44 volts and a power dissipation of 8.17 watts.

Figure 19 shows the schematic for the amplifier. The dc current delivered to the load is mostly supplied by the 15 volt battery attached to the collector of transistor Q7. Transistor Q7 is conducting with the sum of its base and collector currents flowing out of its emitter. Q7's base current of 13 mA is supplied by transistor Q6. As in the previous amplifier, the utility of transistors Q1 through Q3 is unclear.

Input signal inversion and amplification are seen in Figure 20(a) which shows the time domain response to an ac input of 1 microvolt at 1 kHz. The circuit has a flat-band gain of 85.46 dB and a 3 dB bandwidth of 282.8 kHz (Figure 20(b)). The 3 dB bandwidth is significantly better than the previous amplifier. Figure 21 shows the dc transfer characteristic. The dc bias of 5.44 volts can be seen at the voltage input of zero volts. The slope, the magnitude of which is the gain, is negative since the amplifier is inverting the signal.

V. Discussion

We have shown that a linear circuit representation and evolutionary search can automatically produce circuit designs of low to medium difficulty in two applications. Detailed simulations of the evolved designs suggest that all are electrically well behaved and thus suitable for physical implementation. While the evolved designs presented do not exhibit the level of performance of their hand-designed

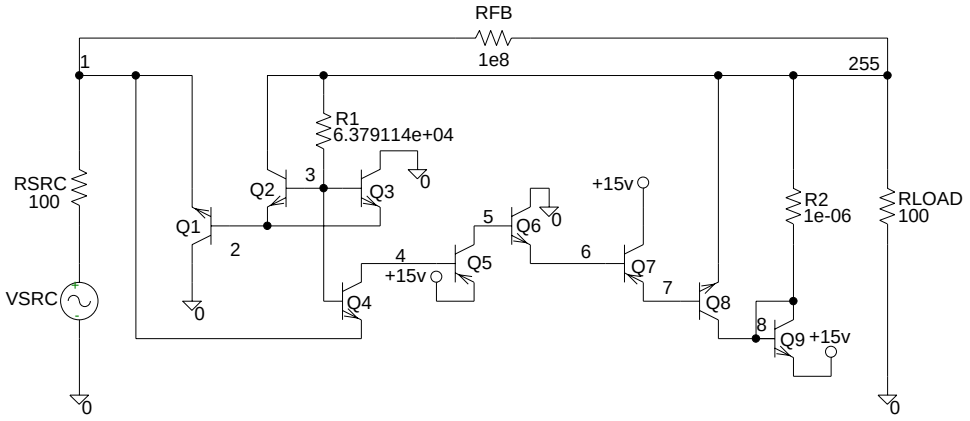

Fig. 16. Circuit schematic of evolved 75 dB amplifier.

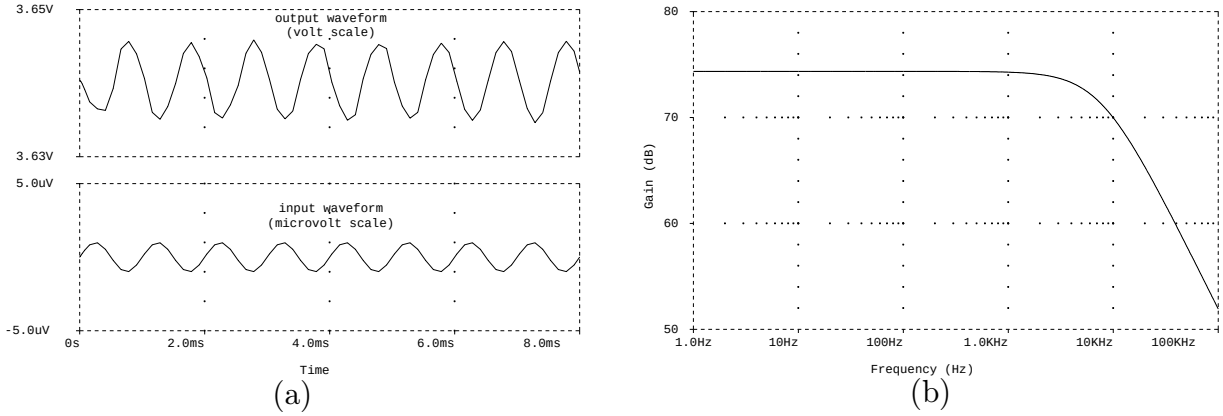

Fig. 17. Small signal behavior of 75 dB evolved amplifier: (a) time domain input waveform is 1 kHz (bottom) which is inverted and amplified (top); (b) frequency response showing 3 dB bandwidth of 7.59 kHz.

predecessors, our main point was to show that a new representation technique with many desirable properties could automatically design practical circuits. The circuit representation method devised permits a wide range of circuits to be constructed, and results in a construction process that is unburdened with repair operations. In addition, the representation is syntactically closed, making it well suited for evolutionary search. For other applications, the instruction set can be easily extended to incorporate other devices not mentioned, such as CMOS transistors. The main limitation of our approach is the inherent restriction on circuit topologies. Such restrictions can be overcome by augmenting the instruction set, and this is one line of investigation we are pursuing. To gain performance on par with circuits designed by engineers, it will be necessary to place further constraints into the fitness functions. For example, practical amplifiers are typically judged by a dozen or so specifications. To evolve an amplifier that would perform as well would require using a multiobjective fitness function that accounts for each specification. This is another area for future work. With the encouraging results of our system, we are optimistic that a subset of analog circuit design tasks may be routinely accomplished by means of evolutionary computation in the future.

References

[1] G. Gielen, W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits,* Boston, MA: Kluwer, 1991.
[2] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, 1989.
[3] J.B. Grimbleby, "Automatic Analogue Network Synthesis using Genetic Algorithms," *Proc. First Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA),* 1995, pp. 53-58.
[4] F. Gruau, "Artificial Cellular Development in Optimization and Compilation," in *Toward Evolvable Hardware* (Lecture Notes in Computer Science), E. Sanchez and M. Tomassini, Eds. vol 1062, Berlin: Springer-Verlag, 1996, pp. 48–75.
[5] R. Harjani, R.A. Rutenbar, L.R. Carey, "A Prototype Framework for Knowledge-Based Analog Circuit Synthesis," *Proc. 24th Design Automation Conf.,* 1987.
[6] J.H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, 1975.
[7] D.H. Horrocks, Y.M.A. Khalifa, "Genetically Derived Filters using Preferred Value Components," *Proc. IEE Colloq. on Linear Analogue Circuits and Systems,* Oxford, UK, 1994.

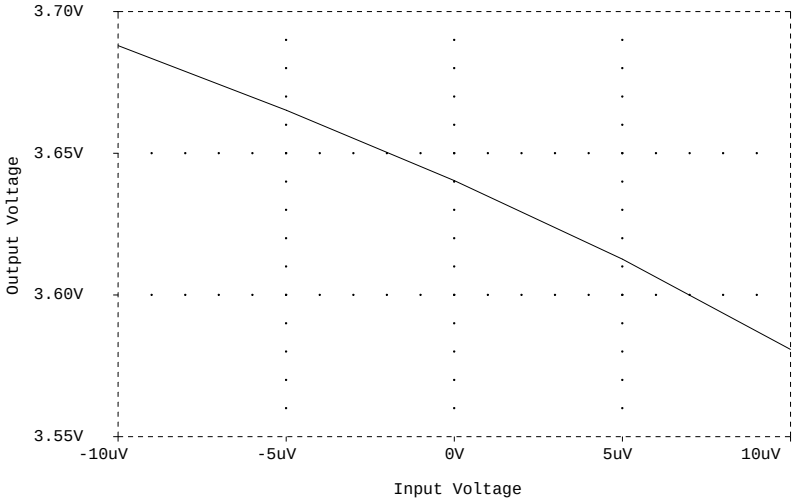

Fig. 18. DC transfer characteristic of 75 dB amplifier.

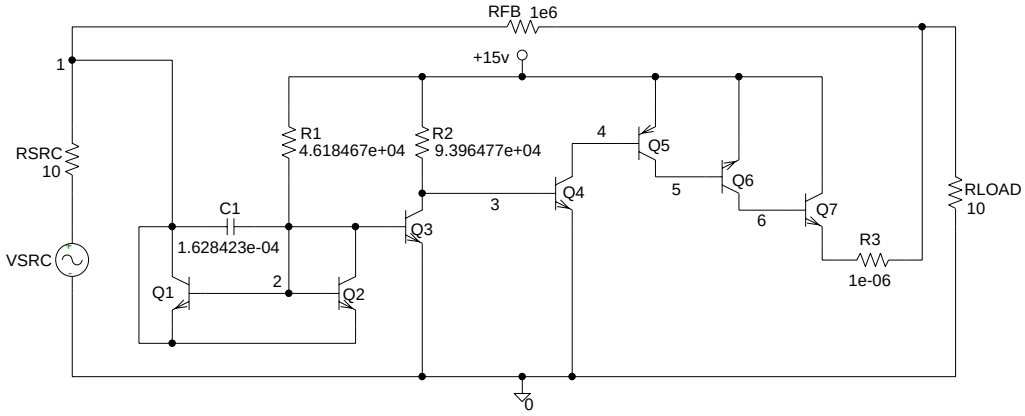

Fig. 19. Circuit schematic of evolved 85 dB amplifier.

[8] L. Huelsbergen, E. Rietman, and R. Slous, "Evolving Oscillators *in Silico*," appears in this issue.
[9] L.P. Huelsman, *Active and Passive Analog Filter Design*, New York: McGraw-Hill, 1993.
[10] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* Cambridge, MA: MIT Press, 1992.
[11] J.R. Koza, F.H. Bennett, D. Andre, M.A. Keane, F. Dunlap, "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 2, July, 1997, pp. 109–128.
[12] J.R. Koza, F.H. Bennett, J.D. Lohn, F. Dunlap, M.A. Keane, D. Andre, "Use of Architecture-Altering Operations to Dynamically Adapt a Three-Way Analog Source Identification Circuit to Accommodate a New Source," in *Genetic Programming 1997 Conference*, J.R. Koza, K.Deb, M.Dorigo, D.B. Fogel, M. Garzon, H. Iba, and R.L. Riolo, (eds), Morgan Kaufmann, 1997, pp. 213–221.
[13] M.W. Kruiskamp, *Analog Design Automation using Genetic Algorithms and Polytopes,* Ph.D. Thesis, Dept. of Elect. Engr., Eindhoven University of Technology, Eindhoven, The Netherlands, 1996.
[14] M. Murakawa, S. Yoshizawa, T. Adachi, S. Suzuki, K. Takasuka, M. Iwata, and T. Higuchi, "[Title TBD]," appears in this issue.
[15] E.S. Ochotta, R.A. Rutenbar, L.R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," *IEEE Trans. Computer-Aided Design,* vol. 15, pp. 273–294, 1996.
[16] T. Quarles, A.R. Newton, D.O. Pederson, A. Sangiovanni-Vincentelli, *SPICE 3 Version 3F5 User's Manual,* Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley, CA, 1994.
[17] A. Stoica, "On Hardware Evolvability and Levels of Granularity," *Proc. 1997 Int. Conf. Intell. Systems and Semiotics,* 1997, pp. 244-247.
[18] G.J. Sussman, R.M. Stallman, "Heuristic Techniques in Computer-Aided Circuit Analysis," *IEEE Trans. Circuits and Systems,* vol. 22, 1975.
[19] R.S. Zebulum, M.A. Pacheco, M. Vellasco, "Comparison of Different Evolutionary Methodologies Applied to Electronic Filter Design," *1998 IEEE Int. Conf. on Evolutionary Computation*, Piscataway, NJ: IEEE Press, 1998, pp. 434–439.

Appendix

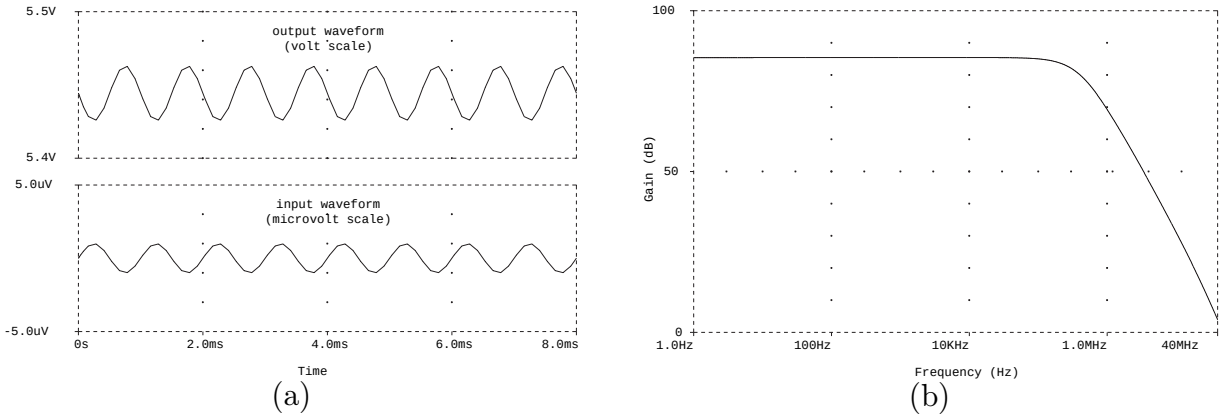

Fig. 20. Small signal behavior of 85 dB evolved amplifier: (a) time domain input waveform is 1 kHz (bottom) which is inverted and amplified (top); (b) frequency response showing a flatband gain of 85.46 dB.

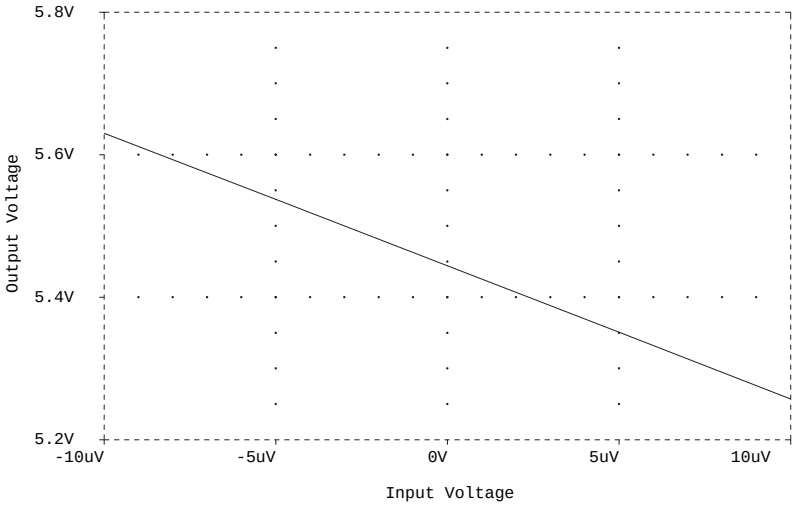

Fig. 21. DC Transfer characteristic of 85 dB amplifier.

```
C_MOVE_TO_NEW C  1  2 10837006
C_MOVE_TO_NEW C  2  3 3547581
C_CAST_TO_PREV C  3  2 9862769
L_CAST_TO_PREV L  3  2 6880711
L_CAST_TO_INPUT L  1  3 7668359
R_CAST_TO_PREV R  3  2 10844497
L_CAST_TO_INPUT L  1  3 6794434
L_CAST_TO_INPUT L  1  3 10915194
L_CAST_TO_INPUT L  1  3 10360067
L_CAST_TO_PREV L  3  2 53267
L_MOVE_TO_NEW L  3  4 7713100
C_MOVE_TO_NEW C  4  5 3568044
C_MOVE_TO_NEW C  5  6 8573596
C_CAST_TO_GND C  0  6 10481
L_CAST_TO_INPUT L  1  6 10897482
L_CAST_TO_INPUT L  1  6 2462113
R_MOVE_TO_NEW R  6  7 12757499
L_CAST_TO_PREV L  7  6 10259
C_CAST_TO_PREV C  7  6 10161281
L_CAST_TO_INPUT L  1  7 2452274
C_CAST_TO_PREV C  7  6 2457158
L_CAST_TO_INPUT L  1  7 15284140
L_CAST_TO_PREV L  7  6 1272684
L_MOVE_TO_NEW L  7  8 11316166
C_CAST_TO_GND C  0  8 24495
L_CAST_TO_PREV L  8  7 15793607
L_CAST_TO_INPUT L  1  8 3300694
L_MOVE_TO_NEW L  8  9 5376858
C_CAST_TO_GND C  0  9 25063
L_CAST_TO_PREV L  9  8 4863538
L_MOVE_TO_NEW L  9 10 4364222
C_CAST_TO_GND C  0 10 12564
L_CAST_TO_PREV L 10  9 4489818
.
```

Fig. 22. Instruction sequence to generate evolved filter number three.

```
Fifth order elliptical lowpass filter
VSOURCE1 256  0  DC 0V  AC 2V
RSOURCE1 256  1  1000
RLOAD001 255  0  1000
CAPC0001    1    2     1.083701e-04
CAPC0002    2    3     3.547581e-05
CAPC0003    3    2     9.862769e-05
RSTR0001    3    2     1.042709e+04
LIND0001    1    3     2.147189e-01
LIND0002    3    2     5.285780e-03
LIND0003    3    4     7.713100e-01
CAPC0004    4    6     2.519509e-05
CAPC0005    0    6     1.048100e-07
LIND0004    1    6     2.008357e-01
RSTR0002    6    7     3.648645e+04
CAPC0006    7    6     1.261844e-04
LIND0005    1    7     2.113217e-01
LIND0006    7    6     1.017696e-03
LIND0007    7    8     1.131617e+00
CAPC0007    0    8     2.449500e-07
LIND0008    8    7     1.579361e+00
LIND0009    1    8     3.300694e-01
LIND0010    8    9     5.376858e-01
CAPC0008    0    9     2.506300e-07
LIND0011    9    8     4.863538e-01
LIND0012    9   10     4.364222e-01
CAPC0009    0   10     1.256400e-07
LIND0013   10    9     4.489818e-01
RSTR0003   10  255     1.000000e-06
*** End netlist
*** Total of   25 components
.AC DEC 20 1 100K
.PRINT AC VM(255)
.END
```

Fig. 23. SPICE circuit netlist for evolved filter number three.